We claim:

1. A method comprising:

providing asynchronous access to multiple users to a graphical programming and analysis environment program;

allowing each user to generate graphically represented code objects within the environment program;

allowing each user access to the code objects of other users based on security privileges accorded to the user;

allowing each user to have the code objects of the user be chained to the code objects of other users to which the user has access to yield inter-code object communication; and,

allowing each user to execute application programs made up of the code objects as chained together within the environment program.

2. The method of claim 1, wherein providing asynchronous access to the multiple users to the graphical programming and analysis environment program comprises enabling multiple users to log into the environment program remotely, such that the multiple users are able to access the environment program simultaneously.

3. The method of claim 1, wherein allowing each user to generate the graphically represented code objects comprises:

allowing each user to instantiate one or more code objects;

allowing each user to determine an internal logic for each code object;

allowing each user to determine first data to be received by each code object; and, allowing each user to determine second data to be sent by each code object.

4. The method of claim 1, wherein allowing each user access to the code objects of the other users based on security privileges accorded to the user comprises rendering visible to each user the code objects of the other users to which the user has access.

5. The method of claim 1, wherein allowing each user to have the code objects of the user to be chained to the code objects of the other users to which the user has access comprises allowing the user to graphically chain code objects together, such that a sender object of a pair of graphically chained together code objects is able to send data that is received by a receiver object of the pair.

6. The method of claim 1, wherein allowing each user to execute the application programs made up of the code objects as chained together within the environment program comprises displaying to the user end results of data processed by the code objects upon execution of the application programs.

7. The method of claim 1, wherein the graphical programming and analysis environment program comprises an applet program, and each code object comprises an applet program within a same applet context as the environment program.

8. The method of claim 7, wherein at least one of the graphical programming and analysis environment program and the code objects is developed within an architecture-

independent and Internet web browsing program-independent computer programming technology.

9. The method of claim 1, wherein the graphically represented code objects coexist with non-graphically represented code objects within the environment program.

10. The method of claim 9, wherein the non-graphically represented code objects comprise stand-alone computer programs.

11. The method of claim 9, wherein the non-graphically represented code objects comprise one or more of: image-viewing programs, video-playing programs, and audio-playing programs.

12. The method of claim 1, wherein the graphically represented code objects comprise one or more of: database objects, video-playing programs, audio-playing programs, image-viewing programs, geo-spatial information map-viewing programs, filter-algorithm programs, and model and analysis tool programs.

13. The method of claim 1, wherein the graphical programming and analysis environment program is visually represented as a white board.

14. The method of claim 13, wherein providing asynchronous access to the graphical programming and analysis environment program comprises providing application programs executable within the white board.

15. The method of claim 14, wherein providing the application programs executable within the white board comprises executing the application programs such that results thereof are immediately available to the multiple users.

16. The method of claim 1, wherein providing asynchronous access to the graphical programming and analysis environment program comprises allowing users to access resources available on a network to which the graphical programming and analysis environment program is communicatively coupled.

17. A multiple-user graphical programming and analysis environment program comprising:

a plurality of graphically represented code objects, each code object created by a user and accessible by other users in accordance with security privileges of the other users;

a plurality of graphically represented inter-code object connections, each inter-code object connection representing data transfer between a pair of code objects;

one or more application programs made up of one or more chains of the code objects interconnected via the inter-code object connections; and,

a graphical white board area within which the code objects are definable and movable and the inter-code object connections are creatable,

wherein the one or more application programs are executable within the graphical white board area.

18. The environment program of claim 17, wherein each code object is an applet program.

19. The environment program of claim 18, wherein the graphical white board area is an applet program having a context within which each code object runs.

20. The environment program of claim 18, wherein the applet program is a Java applet program.

21. The environment program of claim 17, wherein each code object comprises:

a data interface indicating first data to be input into the code object and second data to be output by the code object; and,

internal logic to generate the second data from the first data.

22. The environment program of claim 17, wherein each code object has at least one inter-code object communication graphically terminating on one of an edge and an interior of the code object.

23. The environment program of claim 17, wherein each inter-code object connection represents data being sent by a sender object of the pair of code objects and being received by a receiver object of the pair of code objects.

24. The environment program of claim 23, wherein the data is at least one of: user defined, and filtered according to security privileges accorded to the users.

25. The environment program of claim 17, wherein each inter-code object connection terminates on one of an edge and an interior of one of the code objects.

26. The environment program of claim 17, wherein at least one of the inter-code object connections is one of graphically invisible and purposefully limited in functionality for security.

27. The environment program of claim 17, wherein each inter-code object connection is graphically represented by one of a line and a directed graph.

28. The environment program of claim 17, wherein the one or more application programs are constructed one of asynchronously and synchronously.

29. The environment program of claim 17, wherein the one or more application programs are at least one of: capable of being stored for later retrieval and use, and modular in nature so that more complex application programs may be constructed therefrom.

30. The environment program of claim 17, wherein the one or more application programs are contained within container panels as macro programs, the container panels interconnectable via additional inter-code object connections.

31. The environment program of claim 17, wherein the one or more application programs are at least one of: auditable and loggable during usage, traceable to users who construct the programs, traceable to users who use the programs, and configuration manageable.

32. The environment program of claim 17, wherein the one or more application programs are at least one of: capable of accepting data from dynamically changing input sources,

from static input sources, and from network-accessible resources; capable of network reporting results thereof; and, capable of networking reporting security privilege-filtered results thereof.

33. The environment program of claim 17, further comprising:

a chat area within which the user can communicate with the other users; and,

a user list area showing a name of each of the user and the other users currently logged into the environment program.

34. A method comprising:

accessing by a user a graphical programming and analysis environment program that other users are already currently accessing;

generating by the user graphically represented code objects within the environment program;

graphically chaining together code objects by the user within the environment program, including chaining together the code objects generated by the user and code objects generated by the other users to which the user has access based on security privileges accorded to the user, to yield inter-code object communication; and,

assembling application programs by the user within the environment program, each application program made up of the code objects as have been chained together.

35. The method of claim 34, further comprising executing by the user of the application programs within the environment program.

36. The method of claim 34, wherein generating by the user graphically represented code objects comprises, for each code object,

the user determining a data interface indicating first data to be input into the code object and second data to be output by the code object; and,

the user determining internal logic to generate the second data from the first data.

37. The method of claim 34, wherein chaining together code objects by the user comprises the user, for each pair of code objects to be chained together, specify a sender object of the pair to send data and a receiver object of the pair to receive the data.

38. The method of claim 34, wherein the graphical programming and analysis environment program comprises an applet program, and each code object comprises an applet program within a same applet context as the environment program.